SOLVING A CLASS OF STOCHASTIC MIXED-INTEGER PROGRAMS WITH BRANCH AND PRICE

Eduardo F. Silva
R. Kevin Wood

Operations Research Department
Naval Postgraduate School
Monterey, CA 93943
26 August 2004

ABSTRACT

This paper first describes a class of stochastic mixed-integer programs that have column-oriented formulations suitable for solution by a branch-and-price algorithm (B&P). We then survey several examples, and use a stochastic facility-location problem (SFLP) for a detailed demonstration of modeling and solution techniques. Computational results with a scenario representation of uncertain costs, demands and capacities show that B&P can be orders of magnitude faster than solving the standard formulation by branch and bound; and this can be true even for single-scenario, i.e., deterministic, problems. We also demonstrate how B&P can solve SFLP exactly (as exactly as a deterministic mixed-integer program) when demands and other parameters can be represented as independent, random variables satisfying certain conditions. Our algorithm, based on COIN-OR's open-source code and run under Microsoft Windows on a personal computer, demonstrates just how accessible B&P technology has become.

1 INTRODUCTION

This paper defines a class of stochastic mixed-integer programs (SMIPs) whose instances are amenable to column-oriented formulations, and then shows how to solve such formulations with a branch-and-price algorithm (B&P). The phrase "branch and price" was coined by Savelsbergh (1997), but was first proposed by Johnson (1989) and implemented by Desrochers and Solomon (1989) and Desrochers et al. (1992). The technique combines dynamic column generation, known widely through the "cutting stock problem" (Gilmore and Gomory 1961), with standard branch and bound.

Stochastic programmers have only just begun to see that B&P applies to their problems, and we find only two papers on the topic: Damodaran and Wilhelm (2004) and Lulli and Sen (2004). (However, Shiina and Birge 2004 and Singh et al. 2004 use column-generation without branch and bound to solve SMIPs.) Those papers investigate specific applications of B&P to stochastic programming. In contrast, we describe a complete class of problems to which B&P applies; in similarity, we show impressive computational results.

We begin by describing a general class of mixed-integer, two-stage, stochastic programs whose standard formulations translate easily into column-oriented formulations. We then provide examples, showing how stochastic extensions of several well-known deterministic models fit this framework: the elastic generalized assignment problem (Brown and Graves 1981); crew-scheduling (Vance et al. 1997, Day and Ryan 1997); vehicle-routing problems (e.g., Desrosiers et al. 1995); and the origin-destination integer multicommodity flow problem (Barnhart et al. 2000). One additional problem, a stochastic facility-location problem (SFLP), guides our detailed exploration of the B&P solution approach.

We initially model and solve a version of SFLP with uncertain demands, costs and capacities, all represented through scenarios. Such representations often appear in the stochastic-programming literature (e.g., Butler and Dyer 1999, Chen et al. 2002, Ahmed and Sahinidis 2003, Lulli and Sen 2004). (The primary advantage to a scenario-based representation is that it allows arbitrary dependence among uncertain parameters.) However, another common formulation approach defines individual probability distributions for the stochastic program's parameters; typically, these parameters are assumed independent (e.g., Bertsimas 1992, Zhou and Liu 2003). We will show how B&P can solve SFLP in this situation, too. Furthermore, we will solve the problem exactly, that is, with the same certitude that prevails in deterministic mixed-integer programs. This contrasts with alternative solution procedures that provide only probabilistic or asymptotic guarantees for solution quality (e.g., , Carøe and Tind 1998, Sen and Higle 2000, Ahmed and Sahinidis 2003).

Solution methods for two-stage stochastic programs (TSSPs) with mixed-integer variables and "scenario uncertainty" typically employ Benders decomposition (Benders 1962), or extensions thereof, to the original model. Examples include the integer L-shaped decomposition from Laporte and Louveaux (1993), and the methods developed by Carøe and Tind (1998) and by Sen and Higle (2000). Unfortunately, all of these decompositions use a master problem whose linear-programming relaxation is no stronger than the linear-programming relaxation of the original model (measured over the master-problem variables which the two formulations have in common). Consequently, these decompositions will suffer if the original model formulation has a poor continuous relaxation. In contrast, B&P solves a column-oriented reformulation of a model—also by a form of decomposition—but that reformulation will normally have a much tighter relaxation than the original model.

We construct our B&P algorithm using COIN-OR's open-source code (COIN 2004) coupled with a commercial optimizer, CPLEX (ILOG 2002), to solve the master problem and certain subproblems. A personal computer with a Windows operating system comprises the computing platform. We believe that the use of standard software and hardware like this demonstrates just how accessible B&P technology has become, and we hope this paper helps spur interest in this technology.

The next section describes a particular class of mixed-integer TSSPs and shows how to convert their standard formulations to column-oriented ones. Several models from the literature provide concrete examples. Section 3 presents the SFLP with scenario uncertainty; describes how to solve instances with B&P; and provides computational results. Section 4 presents a version of SFLP where random parameters take on continuous distributions; describes how to solve instances with B&P; and provides computational results. Section 5 investigates some computational enhancements, and section 6 presents conclusions.

2 GENERAL METHODOLOGY

We will show that mixed-integer TSSPs of the following special class are common:

Formulation (TSSP0)

$$\min_{\mathbf{x}} \sum_{i\in I}\left(\mathbf{c}_i\mathbf{x}_i + E_{\tilde{\xi}_i}\left[h_i(\mathbf{x}_i,\tilde{\xi}_i)\right]\right) \tag{1}$$

$$\text{s.t. } \sum_{i\in I} A_i\mathbf{x}_i = \mathbf{b} \tag{2}$$

$$\mathbf{x}_i \in X_i \;\forall i\in I \tag{3}$$

$$\text{where, for all } i\in I,\; h_i(\mathbf{x}_i,\tilde{\xi}_i) = \min_{\mathbf{y}_i} \tilde{\mathbf{f}}_i\mathbf{y}_i \tag{4}$$

$$\text{s.t } \tilde{D}_i\mathbf{y}_i \ge \tilde{B}_i\mathbf{x}_i + \tilde{\mathbf{d}}_i \tag{5}$$

$$\mathbf{y}_i \in Y_i, \tag{6}$$

and where $\tilde{\boldsymbol{\xi}}_i \equiv \text{vec}(\tilde{B}_i, \tilde{D}_i, \tilde{\mathbf{d}}_i, \tilde{\mathbf{f}}_i)$. The sets X_i require all $\mathbf{x}_i$ to be bounded and integral, and the Y_i will normally require, at least, non-negativity of the $\mathbf{y}_i$. The objective-function term $\sum_{i \in I} E_{\tilde{\boldsymbol{\xi}}_i}\left[h_i(\mathbf{x}_i, \tilde{\boldsymbol{\xi}}_i)\right]$ is called the *recourse function*, and matrices $\tilde{D}_i$ are often referred to as *recourse matrices* (Walkup and Wets 1967). We further assume that the model exhibits *relatively complete recourse* (Rockafellar and Wets 1976), which implies that for any $\mathbf{x}_i \in X_i$, an optimal solution $\mathbf{y}_i$, satisfying constraints (5) and (6), can always be found. We note that $\tilde{D}_i = I$ in some of our examples, implying the property of *simple recourse* (Beale 1955, Wets 1966). However, this is not an inherent requirement of this class of problems.

The key feature of TSSP0 is that the recourse function decomposes by "subproblem" *i*.

Because all $\mathbf{x}_i$ are integer, and the sets X_i are bounded, the principles of Dantzig-Wolfe decomposition apply (Dantzig and Wolfe 1960), as extended to integer programming by Appelgren (1969). (See Wolsey 1998, section 11.2, for a comprehensive discussion.) To this end, let $\hat{\mathbf{x}}_i^k \in X_i$, $k \in K_i$, denote the enumerated first-stage solutions for subproblem *i*; because of relatively complete recourse, $E_{\tilde{\boldsymbol{\xi}}_i}\left[h_i(\hat{\mathbf{x}}_i^k, \tilde{\boldsymbol{\xi}}_i)\right]$ is well defined for all such $\hat{\mathbf{x}}_i^k$. Now, because of the special structure, we can embed $E_{\tilde{\boldsymbol{\xi}}_i}[h_i(\hat{\mathbf{x}}_i^k, \tilde{\boldsymbol{\xi}}_i)]$, the decomposed recourse function, into the column costs of a column-oriented formulation for TSSP0:

Column-Oriented, Mixed Integer, Two-Stage Stochastic Program (CTSSP0)

Indices

$i \in I$	subproblems
$k \in K_i$	indices for feasible solutions $\hat{\mathbf{x}}_i^k \in X_i$

Data

$\hat{\mathbf{x}}_i^k$	the k^{th} feasible solution $\hat{\mathbf{x}}_i^k \in X_i$
$\mathbf{c}_i$	first-stage costs for subproblem i

Decision variables

λ_i^k	1 if $\hat{\mathbf{x}}_i^k \in X_i$ is selected, and 0 otherwise;

Formulation (CTSSP0)

$$\min_{\boldsymbol{\lambda}} \sum_{i \in I} \sum_{k \in K_i} \left(\mathbf{c}_i \hat{\mathbf{x}}_i^k + E_{\tilde{\boldsymbol{\xi}}_i}[h(\hat{\mathbf{x}}_i^k, \tilde{\boldsymbol{\xi}}_i)] \right) \lambda_i^k \tag{7}$$

$$\text{s.t.} \quad \sum_{i \in I} \sum_{k \in K_i} \left(A_i \hat{\mathbf{x}}_i^k \right) \lambda_i^k = \mathbf{b} \tag{8}$$

$$\sum_{k \in K_i} \lambda_i^k = 1 \qquad \forall\, i \in I \tag{9}$$

$$\lambda_i^k \in \{0,1\} \quad \forall\, i,k \tag{10}$$

Constraints (9) are often referred to as “convexity constraints.”

CTSSP0 can be applied when first-stage variables are general integers, but binary variables are typical, and we assume this restriction for simplicity. Second-stage variables may be continuous and/or integer.

Naturally, the cardinalities of the index sets K_i may be enormous, and it will usually be necessary to solve CTSSP0 without explicitly enumerating the $\hat{\mathbf{x}}_i^k$. Before discussing such issues, however, we would like to provide examples of how this reformulation technique applies to some problems from the literature. We supply only short descriptions of the problems, and ask to the reader to refer to the references for more details. For simplicity, we hereafter drop the subscript on the expectation operator, because it should be clear from the context.

2.1 **Elastic Generalized Assignment Problem**
(Brown and Graves 1981, Appleget and Wood 2000)

The objective of the *elastic generalized assignment problem* (EGAP) is to minimize the cost of assigning capacity-consuming tasks $j \in J$ to capacitated agents $i \in I$, so that (*i*) each task is assigned to exactly one agent, and (*ii*) the total capacity assigned to agent i does not exceed its (potentially uncertain) capacity $\tilde{u}_i$ unless an appropriate per-unit penalty f_i is paid. If the capacity required by agent i to complete task j is a random variable $\tilde{b}_{ij}$, and the direct cost of that assignment is c_{ij}, then a stochastic version of the EGAP is (Spoerl and Wood 2003):

SEGAP

$$\min_{\mathbf{x}} \sum_{i \in I} \left(\sum_{j \in J} c_{ij} x_{ij} + E\left[h_i \left(\mathbf{x}_i, (\tilde{\mathbf{b}}_i, \tilde{u}_i) \right) \right] \right) \tag{11}$$

$$\text{s.t.} \qquad \sum_{i \in I} x_{ij} = 1 \qquad \forall j \in J \tag{12}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \tag{13}$$

$$\text{where} \quad h_i\left(\mathbf{x}_i, (\tilde{\mathbf{b}}_i, \tilde{u}_i)\right) = \min_{y_i} f_i y_i \tag{14}$$

$$\text{s.t} \quad y_i \geq \sum_{j \in J} \tilde{b}_{ij} x_{ij} - \tilde{u}_i \tag{15}$$

$$y_i \geq 0 . \tag{16}$$

Here, x_{ij} equals 1 if task j is assigned to agent i and is 0 otherwise, and y_i represents agent i's capacity violation. $E\left[h_i\left(\mathbf{x}_i, (\tilde{\mathbf{b}}_i, \tilde{u}_i)\right)\right]$ therefore represents the expected capacity-violation penalty for agent i.

The conversion of SEGAP to a column-oriented formulation is straightforward (Savelsbergh 1997 creates the analogous formulation for a deterministic, inelastic GAP). Each variable represents a potential (joint) assignment of tasks to a particular agent, i.e., a collection of tasks that an agent might be required to perform.

Column-Oriented Formulation for SEGAP (CSEGAP)

Indices

$i \in I$	agents
$j \in J$	tasks
$k \in K_i$	assignment of tasks to agent *i*

Data

$\hat{x}_{ij}^k$	1 if task *j* is assigned to agent *i* in the *k*th assignment of tasks to agent *i*
$\hat{\mathbf{x}}_i^k$	$\left(\hat{x}_{ij_1}^k, \hat{x}_{ij_2}^k, ..., \hat{x}_{ij_{\lvert J \rvert}}^k\right)$, the *k*th assignment vector for agent *i*

X_i	the set of all possible assignments of tasks $\hat{\mathbf{x}}_i^k$ for agent i (The index set K_i can now be completely defined through this relationship: $\bigcup_{k\in K_i} \hat{\mathbf{x}}_i^k = X_i$.)
$\hat{c}_i^k$	expected cost of the kth assignment of tasks to agent i ($\hat{c}_i^k = \mathbf{c}_i\hat{\mathbf{x}}_i^k + E\left[h_i\left(\hat{\mathbf{x}}_i^k, (\tilde{\mathbf{b}}_i, \tilde{u}_i)\right)\right]$ for all $i \in I$ and $k \in K_i$)

Decision variables

λ_i^k	1 if the kth joint assignment of tasks to agent i is chosen, and 0 otherwise;

Formulation (CSEGAP)

$$\min_{\boldsymbol{\lambda}} \sum_{i\in I}\sum_{k\in K_i} \hat{c}_i^k \lambda_i^k \tag{17}$$

$$\text{s.t.} \quad \sum_{i\in I}\sum_{k\in K_i} \hat{x}_{ij}^k \lambda_i^k = 1 \qquad \forall j \in J \tag{18}$$

$$\sum_{k\in K_i} \lambda_i^k = 1 \qquad \forall i \in I \tag{19}$$

$$\lambda_i^k \in \{0,1\} \qquad \forall i \in I, k \in K_i \tag{20}$$

Constraints (18) guarantee that each task j is assigned to exactly one agent, and constraints (19) ensure each agent i receives exactly one assignment of tasks. Note that this simple model allows any task to be assigned to any agent, so X_i consists of all binary vectors of length $|J|$, implying that $|K_i| = 2^{|J|}$ for all i. This model can, indeed, possess a large number of columns.

2.2 Routing and Scheduling with Time Windows
(Desrosiers et al. 1995, Ribeiro and Soumis 1994)

The *vehicle routing problem with time windows* (VRPTW) defines one important exemplar from this problem class. VRPTW describes a fleet of vehicles that must deliver a set of customer orders, with each customer being represented by a node in a network. Vehicles have limited capacities, and customers specify windows of time during which deliveries should be made. The model allocates orders to vehicles so that vehicle capacities are respected, and identifies a route for each vehicle that delivers each order during the customer-specified time window. The column-oriented formulation for this problem looks exactly like CSEGAP, equations (17)-(20), with indices, parameters and variables appropriately redefined, and with capacities probably treated as deterministic quantities. The parameters $\hat{\mathbf{x}}_i^k,\ k \in K_i,$ now represent potential routes (sets of deliveries to customers) for vehicle *i*, each of which covers a subset of the customer set *J*. A probabilistic recourse function could include expected penalties for violating time windows, expected penalties for exceeding maximum route duration, etc.

2.3 Crew Scheduling (Vance et al. 1997, Day and Ryan 1997)

Following the description in Vance et al. (1997), an airline crew-scheduler wishes to minimize the cost of assigning flight crews to a fixed schedule of flights. *Crew pairings* define feasible trip itineraries that can be assigned to some crew. Each pairing consists of a sequence of flights that starts and ends at a home base, respects limits on work hours, allows times for rest breaks, and satisfies numerous other restrictions. Set-partitioning models are the norm for this type of problem (e.g., Vance et al. 1997, Day and Ryan 1997), and these fit a simplified form of CSSTP0 in which *I* is a singleton, **b** is a vector of 1s corresponding to flights that must be covered by crews, and convexity constraints (9) are eliminated; the columns $A_i\hat{\mathbf{x}}_i^k$ represent pairings.

However, “home-base constraints” may need to be enforced (e.g., Butchers et al. 2001) and these simply modify constraints (9) to

$$\sum_{k\in K_i} \lambda_i^k \le u_i \qquad \forall i \in I\ , \tag{21}$$

where I denotes the set of home bases, u_i denotes the number of crews available at $i \in I$, and the index sets K_i now represents potential pairings for crews based at i. For the recourse function, we suggest a probabilistic variant on the function that Ehrgott and Ryan (2002) use to penalize schedules that do not allow adequate time for crews to switch aircraft. Their function is based on averaged historical information, but could be modified to represent a penalty function integrated over empirical or fitted delay distributions.

2.4 **Origin-Destination Integer Multi-Commodity Flow Problem** (Barnhart et al. 2000)

The origin-destination integer multi-commodity flow problem restricts the standard, linear multi-commodity flow problem (Ahuja et al. 1993, chapter 17) by requiring each of a set of commodities to be shipped from its origin to its destination using a single path. The formulation for this problem resembles the well-known path-oriented (i.e., column-oriented) formulation of the linear multi-commodity flow problem (Ford and Fulkerson 1958), but with binary variables. In particular, $\lambda_i^k = 1$ if commodity i follows path $k \in K_i$, and $\lambda_i^k = 0$ otherwise. The main constraints of this problem require that the sum of all commodities flowing across each arc respect that arc’s capacity. Constraints (8), converted to inequalities, handle these requirements if we (*i*) let b_j represent the capacity of arc *j*, and (*ii*) define $a_{ij}\hat{x}_i^k$ to be the amount of arc *j*'s capacity consumed by path *k* of commodity *i*. The convexity constraints (9) guarantee selection of a single path, with appropriate origin and destination, for each commodity. For a communications network, say, each component of the recourse function $E\left[h_i(\mathbf{x}_i,\tilde{\xi}_i)\right]$ might represent an expected, path-dependent penalty based on uncertain link availability (Girard and Sansó 1998) or uncertain "hop delay" (e.g., Papagiannaki et al. 2003) that is independent of congestion.

The following section investigates, in detail, one additional problem that fits the framework of TSSP0 and CTSSP0.

3 SOLVING A STOCHASTIC FACILITY LOCATION PROBLEM BY BRANCH AND PRICE

3.1 A Stochastic Facility Location Problem with Sole Sourcing

A standard, deterministic, facility-location problem aims to identify the best locations for capacitated production facilities that will ship to established customers to meet those customers' demands for some product. The mathematical model must find the best trade-off between variable and fixed costs (Laporte et al. 1994): More open facilities leads to lower shipping (variable) costs because plants are closer to customers, on average; on the other hand, opening more facilities means more facility-installation (fixed) costs are incurred. The deterministic model typically assumes that all customer demands will be completely satisfied, and sometimes requires that each customer be served by a unique facility. This latter requirement is known as *sole-sourcing*, and the resulting model is called the (deterministic) capacitated facility-location problem with sole-sourcing (FLP) (Barcelo and Casanova 1984).

Assume now that some uncertainty in the data arises in the nominally deterministic FLP: Does a manufacturer really know what his demands, capacities and costs will be in the future? Let us represent this uncertainty through a finite, discrete set of scenarios indexed by *s*, with $\mathbf{c}^s$, $\mathbf{d}^s$ and $\mathbf{u}^s$ representing shipping costs, customer demands and facility capacities in each scenario, respectively. For simplicity, we assume that if the aggregate demand for a facility exceeds its capacity to produce, the facility pays a penalty based on the unsupplied amount. This model is reasonable if the "unsatisfied demand" is actually satisfied by the relevant facility acquiring extra product from an outside supplier and shipping it to customers as needed. We are now ready to present a formulation for the SFLP:

Stochastic Facility Location Problem with Sole-Sourcing (SFLP)

Indices

$i \in I$	potential facility locations
$j \in J$	customers
$s \in S$	scenarios

Data [units]

c_i	fixed cost for installing a facility at location *i* [dollars]
$\bar{c}_{ij}$	expected cost to supply all of customer *j*'s demand from facility *i*, assuming no shortfall in facility capacity every occurs [dollars]

d_j^s	customer *j*'s demand under scenario s [tons]
u_i^s	facility *i*'s capacity under scenario s [tons]
f_i^s	penalty for each unit of unmet demand for facility *i* under scenario s [dollars/tons]
p_s	probability that scenario *s* occurs

Decision variables [units]

x_i	1 if facility *i* is opened, and 0 otherwise
x_{ij}	1 if customer *j* is assigned to facility *i*, and 0 otherwise
y_i^s	amount of unmet demand for facility *i* under scenario *s* [tons]

Formulation (SFLP)

$$\min_{\mathbf{x},\mathbf{x}',\mathbf{y}} \sum_{i\in I} c_i x_i + \sum_{i\in I}\sum_{j\in J} \bar{c}_{ij} x_{ij} + \sum_{s\in S}\sum_{i\in I} p_s f_i^s y_i^s \tag{22}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1 \quad \forall j\in J \tag{23}$$

$$-x_i + x_{ij} \le 0 \quad \forall i\in I,\ j\in J \tag{24}$$

$$\sum_{j\in J} d_j^s x_{ij} - y_i^s \le u_i^s \quad \forall i\in I, s\in S \tag{25}$$

$$x_i \in \{0,1\} \quad \forall i\in I \tag{26}$$

$$x_{ij} \in \{0,1\} \quad \forall i\in I,\ j\in J \tag{27}$$

$$y_i^s \ge 0 \quad \forall i\in I, s\in S \tag{28}$$

This type of formulation is known as the *extensive form* of a stochastic program (Birge and Louveaux 1997, p. 8), because the second-stage variables and constraints are made explicit for all scenarios.

3.2 A Column-Oriented Formulation for SFLP

Here we describe a column-oriented formulation for SFLP (CSFLP) that fits directly into the format of CTSSP0. In this formulation, the term *assignment* represents any collection of customers that are served by the same facility. Actually, the forms of CSFLP and CSEGAP are identical, requiring only a redefinition of indices and variables (and definitions from SFLP which will not be repeated). (We note that others have used column generation for solving deterministic facility-location problems before; see Teo and Shu 2004 and Lorena and Senne 2004.)

Column-Oriented Formulation of SFLP (CSFLP)

Indices

$k \in K_i$ possible assignments of customers to a facility i

Data [units]

$\hat{x}_{ij}^{k}$	1 if customer j is assigned to facility i in the kth assignment of customers to that facility
$\hat{c}_i^{k}$	total expected cost of the kth assignment of customers to facility i ($\hat{c}_i^k = c_i + \sum_{j \in J} \overline{c}_{ij} \hat{x}_{ij}^k + \sum_s p_s f_i^s \hat{y}_i^s$, except $\hat{c}_i^k = 0$ for the null assignment) [dollars]

Decision variables

λ_i^k 1 if the k^{th} assignment of customers to facility *i* is chosen, and 0 otherwise

Formulation (CSFLP**)**: Same as (17)-(20)

A column-oriented formulation like CSFLP cannot be solved directly because it is impossible, or impractical, to create the full set of columns. Therefore, each K_i is replaced by a subset to form a *restricted master problem* (RMP). The solution to the LP relaxation of the RMP (LP-RMP) then yields dual variables, which can be used to identify one or more new columns with favorable reduced costs through one or more *column-generation subproblems*. In the case of CSFLP, if we seed the RMP with all null assignments, the following subproblem arises for any facility *i*:

$\mathbf{CSUB}_i(\hat{\boldsymbol{\pi}}, \hat{\mu}_i)$

$$\min_{\mathbf{x}_i, \mathbf{y}_i} \sum_{j \in J} \left(\bar{c}_{ij} - \hat{\pi}_j\right) x_{ij} + \sum_{s \in S} p_s f_i^s y_i^s + c_i - \hat{\mu}_i \tag{29}$$

$$\text{s.t.} \quad \sum_{j \in J} d_j^s x_{ij} - y_i^s \le u_i^s \qquad \forall s \in S \tag{30}$$

$$x_{ij} \in \{0,1\} \qquad \forall j \in J \tag{31}$$

$$y_i^s \ge 0 \qquad \forall s \in S\,, \tag{32}$$

where $\hat{\pi}_j$ is the optimal dual variable associated with constraint (18) for customer *j* in LP-RMP, and $\hat{\mu}_i$ is the optimal dual variable from LP-RMP for the convexity constraint (19) associated with facility *i*. By assuming that all of customer *i*'s demand is shipped from this customer's assigned facility (even if a penalty accrues because the facility's capacity is exceeded), only the expected values of shipment costs, $\bar{c}_{ij}$, need be considered. Thus, if $\tilde{c}'_{ij}$ denotes the random, unit shipping cost from facility *i* to customer *j*, $\bar{c}_{ij} = E[\tilde{c}'_{ij}\tilde{d}_j]$, or $\bar{c}_{ij} = E[\tilde{c}'_{ij}]E_{\xi}[\tilde{d}_j]$ if independence prevails.

If the solution to $\text{CSUB}_i(\hat{\boldsymbol{\pi}}, \hat{\mu}_i)$ defines a non-null assignment of customers to facility *i*, this subproblem's optimal objective-function value gives the reduced cost of the assignment with respect to the current solution of LP-RMP. A negative reduced cost indicates that $\hat{\mathbf{x}}_i^k$ should be translated into a column for the RMP, and inserted into it. If the null assignment is optimal here—recall that the RMP already contains the corresponding column and therefore this assignment cannot be favorable—then no favorable column currently exists for facility *i*.

3.3 Solving the Column-generation Subproblems

The subproblems $\text{CSUB}_i(\hat{\boldsymbol{\pi}}, \hat{\mu}_i)$ are *multi-dimensional knapsack problems* (Weingartner and Ness 1967) with elastic penalties in each dimension; Kleywegt et al. (2002) refer to these as *static stochastic knapsack problems*. We solve them through straightforward branch and bound, except that we add "explicit constraint branching" (Appleget and Wood 2000) by defining the general integer variables g_i and adding the following constraint to each subproblem *i*:

$$\sum_{j \in J} x_{ij} - g_i = 0\,. \tag{33}$$

The variable g_i is an "ECB variable" and receives a higher priority for branching than does any x_{ij}. Intuitively, constraint branching provides a better balanced branch-and-bound enumeration tree, and this tends to reduce total enumeration (see Ryan and Foster 1981).

3.4 Solving the LP-Relaxation of the Master Problem

Branch-and-price algorithms (e.g., Savelsbergh 1997, Barnhart et al. 1998, Silva 2004) are appearing as complements to the branch-and-cut algorithms which can be found implemented in practically all commercial MIP solvers. B&P combines a branch-and-bound algorithm with a column-generation procedure. Achieving good performance with column-generation is difficult (Lübbecke and Desrosiers 2002), but a number of enhancements to the basic procedure can help. “Duals stabilization” comprises the most important enhancement, at least according to our research, so we describe that here briefly. (See du Merle et al. 1999 and Silva 2004 for more detail.)

“Duals stabilization” attempts to accelerate the column-generation process that solves CSFLP’s LP relaxation. We follow Du Merle et al. (1999) for this purpose, and incorporate an *elastic dynamic trust region* for dual variables. The trust region is always centered on the most recent solution. It is elastic because penalized violation of the nominal trust region is allowed, and it is dynamic because its width and penalties are adjusted continually. This trust-region mechanism is implemented by turning master-problem equality constraints into elastic ranged constraints. The primal (master-problem) elastic penalties define the dual trust region’s limits, while primal ranges define the dual penalties, i.e., the penalties applied if the dual variables fall outside the nominal trust region.

A trust region of some sort makes sense in this context because (*i*) the column-generation mechanism, when viewed in the dual, is essentially Benders decomposition

(Benders 1962), and (*ii*) Benders decomposition appears to benefit from the use of trust regions (e.g., Brown et al. 1987, Linderoth and Wright 2002). Of course, many variants on trust regions could be applied to our problems, but this one is simple and has proven effective in recent column-generation experiments (Silva 2004, Singh et al. 2004).

3.5 Computational Results

We implement B&P using software from the COmputational INfrastructure for Operations Research ("COIN-OR," or simply "COIN"), which provides a repository of distinct libraries that can be integrated to build optimization algorithms (Lougee-Heimer 2003). The COIN library labeled "BCP" provides the basic framework for a B&P algorithm (Ralphs and Ladanyi. 2001). Its design anticipates a parallel/distributed environment, and, unfortunately, the protocol that emulates this environment in our serial environment incurs some computational overhead. This overhead could be avoided with some additional programming, so the total solution times reported here, denoted (TT), exclude that overhead. However, we note that the true CPU time for our implementations never exceed TT by more than 10%, and the mean overhead for all problems is only 3.1%.

We have implemented our B&P algorithm using COIN's open solver interface (OSI), coupled with CPLEX 8.0: The linear relaxation of the RMP and the subproblems are submitted to CPLEX's LP solver and MIP solver, respectively. We carry out all tests on a networked workstation, a Dell Dimension 340 with a 2 GHz Pentium 4 processor and 1 GB of RAM. For comparison, we also directly solve the extensive formulations of SFLP using CPLEX 8.0, and report these solution times under "IP" in the tables below.

We investigate eight groups of problems. Each group is defined by problem size, meaning "number of facilities-number of customers," and these sizes are: 5-15, 5-30, 8-24, 8-48, 10-30, 10-40, 10-50 and 10-60. For each problem size, we consider instances with one, ten or fifty scenarios. (Larger problems with more scenarios are considered later.) Because run times vary somewhat between randomly generated instances of the same size, we examine five different instances for each combination of problem size and number of scenarios. All problems in this paper are solved to optimality.

To generate the test problems, we first create a reference problem—the superscript "*R*" below stands for "reference''—according to the following rules: (*i*) Customer demands d_j^R are integers from a discrete uniform distribution *U*(5,25), (*ii*) transportation costs c_{ij}^R are integers from *U*(15,25), (*iii*) facility capacities are $u_i^R = 0.8\sum_{j\in J} d_j^R / |I|$, and (*iv*) the fixed costs are $c_i^R = Cu_i^R$ for some cost-per-unit-capacity conversion constant *C*, which is 1.5 for these examples. Chu and Beasley (1997) use rules (*i*) and (*ii*) to generate the "small instances" of the generalized assignment problem. For our stochastic instances, demands d_j^s are uniformly distributed integers within ±20% of d_j^R, capacities u_i^s are ±10% of u_i^R, and fixed costs are simply $c_i = c_i^R$. Also, facility *i* pays an additional $f_i^s = 0.4\max_{j\in J} c_{ij}^s$ dollars for each unit of demand it must satisfy through an outside purchase. Clearly, the parameter settings we have chosen above are somewhat arbitrary. However, testing assures us that, over a wide range of settings, the large differences in algorithmic performance remain large, and the conclusions reached do not change. We have also experimented with problem generators that correlate costs to

Euclidean distances between facilities and customers that are randomly located on a plane, and also find that our conclusions remain the same.

Tables 1 and 2 show TT for each problem instance solved by (*i*) IP, (*ii*) by basic B&P without duals stabilization and (*iii*) by B&P with duals stabilization. Values in bold indicate the fastest times among the three algorithms. Parameter settings with duals stabilization are fixed for all problems tested, and we set an arbitrary limit of 7,200 seconds on total allowed computation time. Problems are solved to optimality.

Num. of Scenarios	Problem Size (facilities-customers)											
	5-15			5-30			8-24			8-48		
	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz
1	3.5	**0.5**	0.8	2.5	**1.7**	1.7	2274.9	**1.1**	14.5	5.8	5.0	**2.7**
1	**0.1**	0.3	0.8	**1.5**	4.1	3.9	*	**2.0**	2.7	3.8	3.4	**2.4**
1	3.2	**0.6**	2.0	**0.9**	1.4	1.9	274.1	**1.6**	1.8	4.5	6.4	**2.7**
1	**0.2**	0.3	0.6	**1.6**	2.1	1.9	299.8	**1.3**	4.4	3.8	3.2	**2.6**
1	3.6	**0.5**	1.4	8.6	**2.4**	2.8	1.8	**0.8**	0.9	5076.1	61.2	**44.6**
10	1.3	**1.1**	1.6	4.2	**3.7**	3.7	881.0	**4.8**	9.1	142.4	7.9	**4.2**
10	1.4	**1.0**	1.4	**5.1**	16.6	30.5	2987.5	**4.3**	7.0	39.7	6.7	**5.7**
10	1.0	**0.9**	1.1	**1.2**	2.8	3.0	231.3	**4.8**	5.7	20.6	8.4	**5.2**
10	**0.4**	0.6	1.3	**2.4**	3.2	3.6	16.4	**2.7**	2.9	19.8	6.7	**5.9**
10	1.0	**0.7**	1.4	9.4	**3.6**	6.8	3.9	**1.2**	1.3	*	29.7	**25.8**
50	2.5	**2.3**	3.0	**4.9**	10.1	10.2	1637.2	45.5	**29.6**	455.6	**40.0**	58.5
50	3.1	**2.3**	4.4	**11.0**	11.1	11.1	5579.0	8.7	**7.6**	45.7	20.2	**14.0**
50	2.8	**2.3**	2.7	**3.4**	7.8	8.5	1081.6	8.0	**7.6**	53.4	25.0	**15.9**
50	**1.3**	1.4	3.1	**4.3**	8.6	10.1	57.4	**5.8**	6.0	129.1	24.3	**20.9**
50	3.9	**2.7**	3.3	12.2	12.1	**10.7**	6.6	**4.2**	4.2	990.2	**80.1**	114.6

Table legend:

Sc:	Number of scenarios
IP:	CPLEX MIP solver with presolver on
B&P w/o Stz:	Branch-and-price without duals stabilization
B&P w Stz:	Branch-and-price with duals stabilization
*	Problem not solved to optimality within 7,200 CPU seconds.

Table 1. Total time (TT) in CPU seconds, to solve randomly generated SFLPs with scenario uncertainty. Three different algorithms solve five problem instances for each combination of size and number of scenarios. (Note: All generated scenario data for the problem instance in row *r*, for *r* =1,…,10, have been reused in row *r*+5. This accounts for apparent correlations in runtimes as exemplified by rows 5, 10 and 15.) Times marked in bold font are the fastest among the three alternative solution methods.

Num. of Scenarios	Problem Size (facilities-customers)											
	10-30			10-40			10-50			10-60		
	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz	IP	B&P w/o Stz	B&P w/ Stz
1	*	**16.7**	17.0	15.1	3.0	**2.2**	*	**67.5**	143.0	21.5	6.8	**3.8**
1	3.5	**1.1**	1.1	7.8	**2.1**	2.1	*	53.8	**50.8**	22.3	11.0	**5.3**
1	44.8	**1.0**	1.5	14.8	2.8	**2.1**	1657	9.7	**7.2**	14.8	8.3	**4.4**
1	1.4	**1.3**	1.5	*	36.2	**10.2**	*	116.8	**99.1**	13.5	11.9	**5.2**
1	*	**5.3**	19.5	6.2	3.2	**1.7**	50	5.5	**4.1**	23.8	13.4	**4.3**
10	*	**10.3**	12.9	2323.4	6.0	**4.0**	*	20.6	**13.9**	37.2	16.7	**8.0**
10	5.2	2.4	**1.8**	860.1	5.2	**3.8**	*	**310.4**	424.2	3163.1	30.2	**22.4**
10	7.0	3.3	**2.6**	262.6	5.6	**4.1**	*	18.0	**14.1**	140.5	23.1	**14.5**
10	5.0	**2.0**	2.1	*	**53.6**	77.2	*	19.7	**11.7**	45.6	14.2	**10.8**
10	*	**15.0**	29.4	613.5	**5.1**	11.2	*	**20.5**	41.2	30.7	19.7	**9.6**
50	*	16.2	**15.6**	3347.3	20.6	**14.2**	*	**99.1**	127.2	154.6	37.7	**18.8**
50	12.6	**6.5**	7.1	1894.9	14.5	**11.6**	*	37.9	**22.9**	*	60.0	**50.7**
50	51.7	**7.7**	10.7	573.8	14.8	**11.3**	*	**112.3**	171.7	132.3	47.9	**39.9**
50	16.6	6.7	**6.0**	*	30.8	**27.3**	*	33.9	**27.9**	97.1	34.4	**21.5**
50	*	**26.0**	115.0	3559.3	17.3	**12.3**	*	**72.7**	111.5	213.3	39.7	**22.2**

Table legend: same as Table 1

Table 2. Total time (TT) in CPU seconds, to solve randomly generated SFLPs with scenario uncertainty. Three different algorithms solve five problem instances for each combination of size and number of scenarios. This table explores how computation times change as the ratio of facilities to customers decreases. Times marked in bold font are the fastest among the three alternative solution methods.

Num. of Scenarios	Problem Size (facilities-customers)							
	5-15		5-30		8-24		8-48	
	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)
1	18.64	0.00	18.15	0.00	25.29	0.00	22.18	0.00
1	28.48	0.00	22.33	0.00	27.25	0.05	23.26	0.00
1	48.20	0.00	26.65	0.00	27.00	0.00	24.67	0.00
1	16.78	0.04	18.43	0.00	34.65	0.00	22.65	0.00
1	22.16	0.00	19.26	0.00	22.27	0.00	22.49	0.05
10	19.21	0.03	17.11	0.00	25.93	0.07	23.28	0.00
10	34.97	0.00	23.14	0.04	26.21	0.02	21.50	0.00
10	37.36	0.00	24.38	0.00	28.52	0.00	23.79	0.03
10	18.62	0.00	17.77	0.00	31.37	0.00	23.29	0.00
10	20.09	0.00	19.47	0.00	20.93	0.00	21.13	0.00
50	19.37	0.00	17.05	0.00	24.75	0.07	23.66	0.01
50	37.02	0.00	23.15	0.00	25.90	0.00	21.39	0.00
50	36.55	0.00	24.15	0.00	28.68	0.00	24.03	0.02
50	18.87	0.00	17.86	0.00	30.04	0.00	23.73	0.00
50	20.33	0.00	19.22	0.00	21.05	0.00	20.57	0.02

Table 3. Integrality gaps compared between the compact formulation of SFLP (SFLP) and the column-oriented formulation (CSFLP). These results correspond to the problems in Table 1.

Num. of Scenarios	Problem Size (facilities-customers)							
	10-30		10-40		10-50		10-60	
	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)	SFLP (%)	CSFLP (%)
1	22.35	0.04	32.07	0.00	40.43	0.07	20.72	0.00
1	23.00	0.00	26.59	0.00	41.53	0.06	32.15	0.00
1	21.04	0.00	26.16	0.00	56.96	0.02	25.89	0.00
1	23.92	0.00	26.59	0.00	52.95	0.10	23.19	0.00
1	38.46	0.08	27.41	0.00	45.97	0.00	22.71	0.00
10	22.69	0.02	32.30	0.00	37.85	0.02	21.04	0.00
10	22.28	0.00	27.02	0.00	38.12	0.12	35.53	0.01
10	20.32	0.00	25.77	0.03	55.77	0.00	27.42	0.02
10	21.80	0.00	27.55	0.05	48.76	0.04	23.96	0.00
10	42.51	0.09	27.05	0.00	46.41	0.03	21.74	0.00
50	22.66	0.00	33.43	0.00	38.46	0.03	20.21	0.00
50	21.65	0.00	27.64	0.00	36.00	0.00	35.39	0.00
50	20.73	0.00	25.55	0.00	55.82	0.09	26.89	0.00
50	21.82	0.00	28.18	0.00	47.38	0.00	24.36	0.00
50	42.30	0.06	28.03	0.00	44.95	0.06	22.23	0.00

Table 4. Integrality gaps compared between the compact formulation of SFLP (SFLP) and the column-oriented formulation (CSFLP). These results correspond to the problems in Table 2.

3.6 Discussion

Both Tables 1 and 2 provide stark evidence that branch and price can be vastly superior to branch and bound for solving certain stochastic MIPs, and even certain deterministic MIPs as evidenced by the results for the single-scenario problems. The key to this superiority clearly lies in the tighter LP lower bounds provided by CSFLP versus SFLP: See Tables 3 and 4.

One might be concerned that B&P requires so much overhead that it could not be effective for small problems. However, the problems in Table 1 have only five or eight potential facilities, and IP outperforms B&P only in problems with five facilities, and then only by a small amount. Moreover, average solution times for B&P are at least an order of magnitude faster than IP, and IP cannot even solve two of the problems within the time limit of 7,200 CPU seconds. Even for small problems, B&P is a good choice.

Table 2, which covers problems with 10 facilities and 30 to 60 customers, clearly shows that B&P solution times are more stable and suffer less than IP when the number of scenarios increases. Observe that: 23 problem instances out of 60 could not be solved by IP within 7,200 CPU seconds; IP never outperforms B&P; and B&P can be orders of magnitude faster than solving the original problem, even for single-scenario instances, i.e., for deterministic problems.

Table 5, below, explores the computational limits of our current B&P implementation by covering a wider range of problem sizes and number of samples than do Tables 1 and 2. Camm et al. (1997) solve a facility-location model for a commercial application with 17 potential facilities and 123 customer zones, so our largest problem is roughly the same size as at least one real-world problem. We can see here (and to a degree in Tables 1 and 2) that solution times tend to increase only slowly, perhaps linearly, with the number of scenarios. Thus, the number of scenarios does not seem to be a strongly limiting factor with the B&P methodology. This bodes well for solving an SMIP through sampled approximating problems, since the probability of identifying the optimal solution for a discrete TSSP increases exponentially with the number of sampled scenarios (Kleywegt et al. 2002).

Table 5 shows that problem size, in terms of facilities and customers, is a stronger limiting factor in solving SFLP by B&P. We discuss potential reasons for this in the following sections.

Num. of scenarios	Problem Size (facilities-customers)			
	10-60	20-60	15-80	20-100
50	36.4	64.6	47.2	137.2
50	35.2	55.7	65.1	257.2
50	26.9	58.1	54.6	108.2
50	35.9	62.1	44.0	106.0
50	26.7	53.4	106.6	154.9
100	48.2	136.8	132.7	229.6
100	68.4	96.5	108.0	829.6
100	56.2	109.4	238.3	165.7
100	73.0	123.9	75.6	150.0
100	51.0	90.4	76.3	257.0
200	134.6	245.3	181.5	513.0
200	131.3	199.4	302.2	1294.6
200	103.7	168.9	762.1	260.7
200	134.0	184.5	157.1	272.2
200	103.8	174.4	156.5	555.0
300	154.8	374.4	305.4	507.8
300	248.7	305.8	391.2	817.0
300	144.6	250.0	654.1	535.4
300	164.0	320.9	493.2	493.2
300	210.8	276.0	274.4	687.1

Table 5. The total time (TT) in CPU seconds, for randomly generated SFLPs with scenario uncertainty. This table explores the computational limits of our current B&P implementation with duals stabilization.

4 SOLVING A SPECIAL CASE OF SFLP EXACTLY

Here we investigate a special case of the SFLP in which uncertain parameters are independent, continuously distributed random variables. This model paradigm appears frequently in the literature (e.g., Louveaux and Peeters 1992, Laporte et al. 1994); however, such models are rarely solved exactly as we shall solve SFLP. Even if the reader believes such assumptions are unreasonable in a real-world facility-location problem—independence of demands seem particularly unlikely in the SFLP, for instance—it is instructive to see that exact solutions can be achieved for such a model in the column-oriented framework. Perhaps these assumptions will be more appropriate in other applications of our methodology.

Consider now a random vector $\tilde{\boldsymbol{\xi}} = \text{vec}(\tilde{\mathbf{c}}, \tilde{\mathbf{d}}, \tilde{\mathbf{f}})$ whose elements represent shipping costs, customer demands and unmet-demand penalties, respectively. The column-oriented formulation for SFLP with these random parameters resembles equations (17)-(20), with the following definitions for the data:

Data [units]

c_i	fixed cost for installing a facility at location *i* [dollars]
$\tilde{c}_{ij}$	unit shipping cost from facility *i* to customer *j* [dollars/ton]
$\tilde{d}_j$	demand from customer *j* [tons]
$\bar{c}_{ij}$	expected cost for supplying all of customer *j*'s demand from facility *i*, i.e., $\bar{c}_{ij} = E[\tilde{c}_{ij}\tilde{d}_{ij}]$ [dollars]
u_i	capacity of facility *i* [tons]
$\tilde{f}_i$	unit penalty for unmet demand that must be covered by facility *i* [dollars/tons]
$\bar{f}_i$	expected unit penalty $E[\tilde{f}_i]$ [dollars/tons]
$\hat{c}_i^k$	total expected cost of the k^{th} assignment of customers to facility *i*, $\hat{c}_i^k = \bar{\mathbf{c}}_i \hat{\mathbf{x}}_i^k + E_{\tilde{\xi}_i}[h_i(\hat{\mathbf{x}}_i^k, \tilde{\boldsymbol{\xi}}_i)])$ [dollars] or, more precisely,

$$\hat{c}_i^k = \begin{cases} 0 & \text{if } \hat{\mathbf{x}}_i^k = 0 \\ \overline{\mathbf{c}}_i \hat{\mathbf{x}}_i^k + \overline{f}_i E\left[\left(\sum_{j \in J} \tilde{d}_j \hat{x}_{ij}^k - u_i\right)^+\right] + c_i & \text{otherwise} \end{cases} \tag{34}$$

4.1 Normally Distributed Demands

For the special-case model, demand for customer *j* is independent of other random quantities, and is assumed to be normally distributed with mean m_j and variance v_j, i.e., $\tilde{d}_j \sim N(m_j, v_j)$. Unmet-demand penalties are continuously distributed random variables that are independent of demands and other random parameters. In fact, under independence, these penalties appear in the objective function only through their means, so they may have arbitrary distributions with finite means. Consequently, we represent these penalties through their vector of means, denoted $\overline{\mathbf{f}}$.

For simplicity in exposition, we also assume that the mean and variance for each demand $\tilde{d}_j$ are integers. The reader will see that our techniques easily extend to means $\alpha_j m_j$ and variances $\beta_j v_j$, where α_j and β_j are positive scale parameters, and m_j and v_j represent integers running from 0 to some finite upper bound. Given the efficiency of the dynamic-programming solution procedure that uses m_j and v_j, the scale parameters can be quite small, and thus a wide range of actual mean-and-variance combinations can be closely approximated.

The RMP for this model does not change from the column-oriented formulation (CSFLP) presented in section 3.2. To solve this special case exactly, we will exactly solve the subproblems corresponding to the formulation (29)-(32). Given equation (34), the subproblem associated with facility i is this static stochastic knapsack problem (Kleywegt et al. 2002):

$$z^* = \min_{x_{ij}\in\{0,1\}\ \forall j\in J} \left\{ \sum_{j\in J} \left(\bar{c}_{ij} - \pi_j\right) x_{ij} + \bar{f}_i E\left[\left(\sum_{j\in J} \tilde{d}_j x_{ij} - u_i \right)^+ \right] \right\} + c_i - \mu_i \tag{35}$$

where $s^+ \equiv \max\{0, s\}$. Evaluating $E\left[\left(\sum_{j\in J} \tilde{d}_j x_{ij} - u_i \right)^+ \right]$ is easy, because we know that (Kleywegt et al. 2002)

$$E\left[\tilde{w}(m,v)^+ \right] = m\Phi\left(\frac{m}{\sqrt{v}} \right) + \sqrt{\frac{v}{2\pi}} \exp\left(-\frac{m^2}{2v} \right), \tag{36}$$

for any $\tilde{w}(m,v) \sim N(m,v)$, where $\Phi(\bullet)$ denotes the cumulative distribution function of a standard normal random variable.

Ignoring the constraint-violation penalties for the moment, we apply dynamic programming to evaluate the functions $g_i(|J|, m_i, v_i)$ defined as

$$g_i(|J|, m_i, v_i) = \min \sum_{j\in J} (\bar{c}_{ij} - \pi_j) x_{ij} \tag{37}$$

$$\text{s.t.} \qquad \sum_{j\in J} m_j x_{ij} \le m \tag{38}$$

$$\sum_{j\in J} v_j x_{ij} = v \tag{39}$$

$$x_{ij} \in \{0,1\} \tag{40}$$

for $m = 0,\dots,m_{\max}$, and $v = 0,\dots,v_{\max}$, where $m_{\max} = \sum_{j\in J} m_j$ and $v_{\max} = \sum_{j\in J} v_j$. (In practice, much smaller limits on $m_{\max}$ and $v_{\max}$ can and should be used for the sake of efficiency.)

Initialization:

$g_i(j,m,v) = 0$ for $j = 0, m = 0, v = 0;$

$g_i(j,m,v) = +\infty$ for $j = 0, m \neq 0, v \neq 0;$

Recursion:

$$g_i(j,m,v) = \min_{\substack{j=1,\dots,|J| \\ m=1,\dots,m_{\max} \\ v=1,\dots,v_{\max}}} \left\{ g_i(j-1,m,v), \bar{c}_{ij} - \pi_j + g_i(j-1, m-m_j, v-v_j) \right\} \tag{41}$$

This recursion is similar to that for a two-dimensional knapsack problem, but for a given m, the objective value g_i does not depend on the index v. This variance will be used in a final calculation, however.

Now, since $\hat{\mathbf{x}}_i$, an assignment of customers to a facility i, yields an aggregate demand with distribution $N\left(\sum_{j\in J} m_j \hat{x}_{ij}, \sum_{j\in J} v_j \hat{x}_{ij} \right)$, the optimal objective value for (35) will be

$$z^* = \min_{\substack{m=1,\dots,m_{\max} \\ v_{\max}=1,\dots,v_{\max}}} \left\{ g\left(|J|, m, v\right) + \bar{p}_i E\left[\tilde{w}(m-u_i, v)^+ \right] \right\} + c_i - \mu_i . \tag{42}$$

For the case where the facilities capacities $\tilde{u}_i$ are also independent and normally distributed, and independent from the customer demands, the method just described will work after making a single modification: The expectation in equation (42) changes to $E\left[\tilde{w}\left(m-E[\tilde{u}_i],v+Var(\tilde{u}_i)\right)^+\right]$.

4.2 Extensions

The methodology described above will fit other problems, but numerical integration may be required. Suppose, for instance, that each demand can be defined by $\tilde{d}_j=\sum_{k=1}^{K_j}\tilde{r}_{jk}+m_j$ where all $\tilde{r}_{jk}$ independent and identically distributed (iid) with mean 0 and variance *v*, and all m_j are positive integers; all u_i are deterministic, here. Then, any aggregate demand $\sum_{j\in J}\tilde{d}_j x_{ij}$ can be described through its integer mean $m'=\sum_{j\in J}m_j x_{ij}$, and its scaled integer variance $v'=v\sum_{j\in J}K_j$. Thus, the recursion (41) applies with appropriate adjustments for the scaled variances, and $E\left[\left(\sum_{j\in J}\tilde{d}_j x_{ij}-u_i\right)^+\right]$ can be computed by numerical integration over the distribution of $\sum_{j\in J}\tilde{d}_j x_{ij}$. This will be straightforward since that distribution is defined through the mean-shifted convolution of $t=\sum_{j\in J}x_{ij}$ iid random variables, which is completely defined by its bounded integer mean m', its bounded scaled-integer variance v', and the common distribution for $\tilde{r}_{jk}$.

4.3 Computational Results for Normally Distributed Demands

To build test instances for this special case, we select each of the single-scenario problems from section 3, and assume that its demand represents the expected value of a normally distributed demand. The variance for each such demand is then generated as a discrete uniform random variable on [1, V_j], where V_j is the maximum value that assures $\mathrm{P}(\tilde{d}_j < 0) \le 0.001$ (e.g., Spoerl and Wood 2003). Table 6 displays solution times and integrality gaps for all 40 problem instances. We use the software suite of section 3 for solving these problems, but the computer is an IBM G40, Pentium 4 laptop computer with 1 GB of RAM, running at 3 GHz.

Problem Size		B&P without duals stabilization			B&P with duals stabilization			Int. gap (%)
Facilities	Customers	Soln. time (TT, sec.)	Cols.	Nodes	Soln. time (TT, sec.)	Cols.	Nodes	
		0.8	188	1	**0.4**	194	13	0.00
		1.0	153	1	**0.5**	181	1	0.00
5	15	0.5	171	1	**0.4**	143	1	0.00
		0.6	192	1	0.7	205	1	0.00
		0.7	168	1	**0.6**	191	1	0.00
		11.3	443	1	**9.7**	404	1	0.00
		9.3	474	1	**7.3**	437	1	0.00
5	30	9.3	432	1	**8.8**	438	1	0.01
		9.7	423	1	**8.4**	416	1	0.00
		18.8	599	9	**17.5**	695	7	0.04
		1.2	350	1	**1.1**	337	1	0.00
		1.1	322	1	**0.9**	310	1	0.00
8	24	3.0	466	9	**1.7**	326	3	0.05
		1.2	351	1	**0.8**	296	1	0.00
		1.8	385	1	**1.6**	363	1	0.00
		9.1	1001	1	**6.4**	778	1	0.00
		10.2	1006	1	**7.0**	765	1	0.01
8	48	**11.2**	1166	5	11.5	1237	9	0.02
		10.4	989	3	**7.3**	816	9	0.01
		11.5	959	3	**7.9**	734	3	0.00
		3.3	512	5	**2.9**	455	7	0.01
		2.8	501	1	**2.3**	452	1	0.02
10	30	2.7	524	1	**2.0**	451	1	0.00
		3.3	526	1	**2.4**	461	1	0.01
		2.5	497	3	5.7	856	3	0.03
		6.5	823	1	**5.0**	708	3	0.00
		9.3	860	1	**6.5**	679	1	0.00
10	40	13.9	793	3	**11.5**	668	3	0.02
		14.0	893	13	19.8	1326	5	0.03
		10.0	791	1	**6.7**	662	3	0.01
		28.5	1076	1	**20.7**	862	3	0.03
		37.5	1180	5	**22.9**	794	5	0.01
10	50	**48.9**	1624	17	65.3	2144	19	0.07
		32.2	1303	5	34.1	1423	5	0.02
		24.5	1105	1	**17.9**	837	1	0.00
		73.4	1357	1	**56.1**	1221	1	0.00
		76.1	1585	15	**46.2**	1263	5	0.01
10	60	71.7	1434	1	**50.1**	1213	1	0.00
		90.7	1581	5	**66.9**	1243	5	0.00
		74.0	1418	1	**55.6**	1215	1	0.00

Table 6. Total time (CPU seconds) to solve SFLP with B&P when demands are independent and normally distributed. The times in bold font indicate the fastest solution time for each problem. (Pentium 4, 3 GHz computer with 1 GB of RAM.)

Problem Size (facilities-customers)											
10-60		20-60		15-80		20-100		25-150		30-200	
Soln. time (TT, sec.)	Cols.	Soln. time (TT, sec.)	Cols.	Soln. time (TT, sec.)	Cols.	Soln. time (TT, sec.)	Cols.	Soln. time (TT, sec.)	Cols.	Soln. time (TT, sec.)	Cols.
56.1	1221	9.4	1039	110.9	1947	180.2	2421	323.2	3753	1799.7	5859
46.2	1263	9.3	1068	80.5	1719	253.6	3144	451.5	4051	2386.7	6776
50.1	1213	72.9	4092	73.8	1683	117.8	2284	1014.4	5057	*	6644
66.9	1243	11.6	1080	91.4	2023	153.0	2355	613.6	4025	*	6569
55.6	1215	29.4	2525	65.6	1585	208.9	2631	1406.7	5358	2337.7	6359

Table 7. Larger instances of SFLP. Total time (CPU seconds) to solve SFLP with B&P, duals stabilization only, when demands are independent and normally distributed. Times marked as "*" indicate the problem could not be solved in 2400 seconds. (Pentium 4, 3 GHz computer with 1 GB of RAM.)

4.4 Discussion

As in the Section 3, we see that duals stabilization is a useful enhancement to the B&P algorithm.

With some exceptions, results displayed in Tables 1, 2, 5 and 6 indicate that B&P performs better on problems with a smaller facilities-to-customers ratio. Similar results have been observed when solving generalized assignment problems, where the tasks-to-agents ratio correlates positively with the number of feasible solutions the problem instances have (Savelsbergh 1997, Silva 2004). In turn, the number of feasible solutions is positively correlated with the number of columns in the column-oriented model, and the more columns a problem has, the harder it must be to solve using B&P. The glaring contradiction to this argument is the 10-60 column in Table 2, which shows that these problems are easier than the 10-50 problems. Theoretically, these 10-60 problems may have more columns than the 10-50 problems, but the effective number, i.e., the number of "cost-effective columns" may be smaller. Note that Table 4 shows that the extensive 10-60 models have tighter LP relaxations than do their 10-50 counterparts, which implies the system is more capacity-bound in some sense (an artifact of the problem generator). This, in turn, may mean that the only cost-effective columns are those that use most of a facility's nominal capacity, and this is a relatively small number.

5 OTHER COMPUTATIONAL ENHANCEMENTS

The purpose of this paper is to describe an entirely new technique for solving a class of SMIPs, not to explore a wide range of computational enhancements for this technique. We have found that one enhancement, duals stabilization, is important for good performance, so we have provided detailed computational results to demonstrate that fact. However, we have begun preliminary exploration of a number of potential computational enhancements, and believe that this warrants a brief discussion. Other researchers may wish to explore these and other potential enhancements in detail, along with their myriad parameter settings. We explore the following potential enhancements:

1. "Strong branching:" A set of variables that appear attractive for branching purposes, rather than a singleton, is selected, and branching is carried out for each variable selected. Child problems for each branch are created, and fully optimized, and the most attractive branch is followed. "Most attractive" simply implies the branch with minimum objective function value for our implementation, but other rules could be used, of course.. We only consider "strong-branching sets" of cardinality two: This is a small number compared to standard applications of strong branching, but the B&P paradigm requires much more work to re-optimize after branching, so this number must be kept small.

2. “Solve one subproblem:” Rather than attempting to generate a favorable column for each subproblem before returning to the master problem, we return to the master problem as soon as some subproblem generates a favorable column: The column is added to RMP; the LP-RMP is re-solved; the order of the subproblems is randomized to ensure that algorithm does not focus on one subproblem at the expense of others; and the search for favorable columns continues. (Tests show this scheme can be an order of magnitude faster than scanning the subproblems in a fixed order, and substantially faster then randomizing only after scanning all subproblems in a temporarily fixed order.)
3. “Delete poor columns:” Periodically, all columns with reduced cost above a given threshold are deleted from the RMP.

Intuitively, strong branching simply extracts extra information from the enumeration tree, at some computational cost. Is that cost worth paying? In many problems, little or no branching takes place, so strong branching cannot help, but it also incurs little or no cost then.

“Solve one subproblem” is potentially attractive because “solve all subproblems,” the standard approach, generates a set of columns based on a common set of dual variables. Because of this, the resulting columns may exhibit much overlap in terms of the customers assigned. But, overlapping columns cannot exist in an optimal solution to SFLP, so much column-generation effort may be wasted.

The analog of “delete poor columns” is commonly used in the dual to column generation, Benders decomposition (Alvarez 2004, Brown 2004). Benders decomposition repeatedly generates and adds constraints to a master problem, and it may be beneficial to limit the size of the master problem by periodically eliminating non-binding constraints. Of course, something that is non-binding now may become binding later and a deleted row—column in our case—will need to be regenerated later at some computational cost.

Table 7 displays the results of our preliminary exploration of these "other computational enhancements" on SFLP with normally distributed demands. (Experiments with SFLP under scenario uncertainty demonstrate similar results.) B&P with duals stabilization forms the baseline for computational comparisons, and we only consider the other enhancements combined individually with the baseline. Every potential enhancement improves computation times for at least a few problems. Thus, further investigation of these techniques, combined in various ways and with various parameter settings, seems warranted. However, the overwhelming improvements with "solve one subproblem" seem to indicate that it should be immediately adopted as a standard, along with duals stabilization.

Problem Size		Baseline: B&P w/ duals stabilization	Baseline + strong branching	Baseline + solve one subproblem	Baseline + delete poor columns
Facilities	Customers				
		0.4	0.4	0.5	0.4
		0.5	0.5	**0.4**	0.5
5	15	0.4	0.3	**0.2**	0.4
		0.7	0.7	**0.6**	0.7
		0.6	**0.5**	0.6	0.5
		9.7	9.6	**7.1**	9.3
		7.3	7.2	**5.7**	7.1
5	30	8.8	8.7	**6.7**	8.9
		8.4	8.4	**5.7**	8.7
		17.5	17.2	**11.8**	17.4
		1.1	1.1	**0.9**	1.1
		0.9	1.0	**0.9**	1.0
8	24	1.7	1.7	**1.7**	1.7
		0.8	0.9	0.9	0.8
		1.6	1.5	**1.2**	1.6
		6.4	6.2	**6.0**	6.2
		7.0	6.9	**6.8**	6.8
8	48	11.5	11.6	**8.1**	11.5
		7.3	7.3	**7.1**	7.2
		7.9	11.8	**7.1**	7.9
		2.9	2.4	**1.8**	2.6
		2.3	2.5	**1.7**	2.3
10	30	2.0	2.2	**1.8**	2.0
		2.4	2.5	**1.7**	2.3
		5.7	4.7	**1.7**	4.7
		5.0	5.1	**4.1**	4.9
		6.5	6.9	**5.0**	8.0
10	40	11.5	10.8	**6.3**	11.0
		19.8	20.0	**9.8**	19.8
		6.7	7.1	**5.8**	6.8
		20.7	19.9	**15.7**	23.1
		22.9	38.3	**18.0**	23.1
10	50	65.3	57.0	**32.2**	62.7
		34.1	31.3	85.0	**28.1**
		17.9	18.2	21.9	17.8
		56.1	56.3	**46.8**	55.7
		46.2	46.6	**34.1**	44.3
10	60	50.1	**46.5**	70.2	46.8
		66.9	97.0	**63.2**	71.1
		55.6	57.0	**55.1**	57.4

Table 8. The total time (TT) in CPU seconds, for solving SFLPs with normally distributed demands. This table starts with B&P with duals stabilization as a baseline, and explores the use of other potential computational enhancements. Numbers in bold are the fastest times. "Solve one subproblem" yields the best single improvement, but note that every potential enhancement is substantially better than the baseline for at least one problem.

6 SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

6.1 Summary

This paper has proposed a column-oriented model for a class of two-stage stochastic mixed-integer programs (SMIPs), and has described examples of well-known deterministic optimization problems whose stochastic versions fall into this class. We show how to solve such problems with a branch-and-price algorithm (B&P), using a stochastic facility-location problem (SFLP) as an example. We solve one version with scenario uncertainty as well as one with continuously distributed parameters satisfying certain conditions. We solve both versions exactly, and demonstrate how the algorithm's performance can be improved by "duals stabilization" and other techniques. The open-source code libraries of the COmputational INfrastructure for Operations Research (COIN-OR) provide the framework for our B&P algorithm, while CPLEX 8.0 comprises the solver engine.

6.2 Conclusions

This research demonstrates that B&P is an attractive method to solve certain SMIPs. For SFLP with scenario uncertainty, B&P can be orders of magnitude faster than solving the original problem by branch and bound, and this can be true even for deterministic, i.e., single-scenario problems. And, the ability to solve exactly an SMIP with continuously distributed parameters is highly unusual in the stochastic-programming literature. Finally, we have shown that the COIN/BCP software will run successful under

Microsoft Windows. We believe this should make the B&P methodology more widely accessible to researchers and commercial users.

6.3 Recommendations for Further Work

The B&P approach can be used to solve, at least approximately, SMIPs of the class described in section 2, but with more general probability distributions. For instance, the methods of “sample-average approximations” (Mak et. al 1999, Kleywegt et al. 2002) provide probabilistic guarantees on solution quality and are based on repeated solutions of sampled approximating problems. However, a sampled approximating problem is essentially identical to a stochastic program with scenario uncertainty.

Sampled subproblems can be used to identify favorable columns in a “nearly exact algorithm,” too. Suppose that once a subproblem’s integer variables are fixed, i.e., a column of the model has been defined, the expected cost of that column can be estimated highly accurately through sampling. This certainly holds for SFLP, where the penalties associated with, say, 10,000 sampled demands for some fixed customers-to-facility assignment can be sampled and averaged in a fraction of a second. For all intents and purposes then, that average will exactly equal the expected cost for the column, and the LP-relaxation of a master problem containing such columns would yield exact dual solutions. The only theoretical gap in this procedure is that the solution of a sampled subproblem might indicate that a column is favorable, but extended sampling would reveal that it is not. If we solve many sampled subproblems and cannot identify an improving column, then we might become convinced that we have, in fact, solved the

LP-RMP. However, a formal procedure will need to be constructed to provide a rigorous "level of conviction."

Finally, we note that the COIN/BCP software in originally intended for use in a distributed/parallel environment. It will be interesting to investigate how well our procedures perform in such an environment.

7 ACKNOWLEDGEMENTS

Kevin Wood thanks the Office of Naval Research and the Air Force Office of Scientific Research for their research support, as well as the Naval Postgraduate School and the University of Auckland for their general support. Eduardo Silva thanks the Naval Postgraduate School and the Brazilian Navy for their support. Both authors thank the COIN-OR team for their contributions that made this research possible.

8 REFERENCES

Ahmed, S., N. V. Sahinidis. 2003. An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research* **51** 461-471.

Alvarez, R. E. 2004. Interdicting electrical power grids. Masters Thesis, Operations Research Department, Naval Postgraduate School, Monterey, California.

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows.* Prentice Hall, Englewood Cliffs, New Jersey.

Appleget, J. A., R. K. Wood. 2000. Explicit-constraint branching for solving mixed-integer programs. M. Laguna, J.L. González-Velarde, eds. *Computing Tools for*

Modeling, Optimization and Simulation, Kluwer Academic Publishers, Boston, MA. 243–261.

Appelgren, L. H. 1969. A column generation algorithm for a ship scheduling problem. *Transportation Science* **3**, 53-68.

Barcelo J., J. Casanova. 1984. A heuristic lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research* **15** 212–226.

Barnhart, C., C. A. Hane, P. H. Vance. 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research* **48** 318-326.

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* **46** 316-329.

Beale, E.M.L. 1955. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society*. **17B** 173–184.

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4** 238-252.

Bertsimas, D. J. 1992. A vehicle routing problem with stochastic demand. *Operations Research* **40** 574-585.

Birge, J. R., F. Louveaux. 1997. *Introduction to Stochastic Programming*. Springer-Verlag, New York.

Brown, G. G. 2004. Personal communication. 17 June.

Brown, G. G., G. Graves. 1981. Real-time dispatch of petroleum tank trucks. *Management Science* **27** 19-32.

Brown, G. G., G. Graves, M. Honczarenko. Design and operation of a multicommodity production/distribution system using primal goal decomposition. *Management Science* **33** 1987 1469-1480.

Butchers, E. R., P. R. Day, A. P. Goldie, S. Miller, J. A. Meyer, D. M. Ryan, A. C. Scott, C. A. Wallace. 2001. Optimized Crew Scheduling at Air New Zealand. *Interfaces* **31** 30–56.

Butler, J. C., J. S. Dyer. 1999. Optimizing natural gas flows with linear programming and scenarios. *Decision Sciences* **30** 563-580.

Camm J. D., T. E. Chorman, F. A. Dill, J. R. Evans, D. J. Sweeney, G. W. Wegryn. 1997. Blending OR/MS, judgment, and GIS: Restructuring P&G's supply chain. *Interfaces* **27 (1)** 128-142.

Carøe, C. C., J. Tind. 1998. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming* **83** 451-464.

Chen, Z.-L., S. Li, D. Tirupati. 2002. A scenario-based stochastic programming approach for technology and capacity planning. *Computers and Operations Research* **29** 781-806.

Chu P. C., J. E. Beasley. 1997. A genetic algorithm for the generalized assignment problem. *Computers and Operations Research* **24** 17-23.

COIN. 2004. http://www.coin-or.org (accessed July 2004).

Day, P. R., D. M. Ryan. 1997. Flight attendant rostering for short-haul airline operations. *Operations Research* **45** 649-661.

Damodaran, P., W. E. Wilhelm. 2004. Branch-and-price methods for prescribing profitable upgrades of high-technology products with stochastic demands. *Decision Sciences* **35** 55-82.

Dantzig, G. B., P. Wolfe. 1960. The decomposition principle for linear programs. *Operations Research* **8** 101-111.

Desrochers, M., J. Desrosiers, M. Solomon. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* **40** 342-354.

Desrochers, M., F. M. Solomon. 1989. A column generation approach to the urban transit crew scheduling problem. *Transportation Science* **23** 1-13.

Desrosiers J., Y. Dumas, M. M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser eds. *Handbooks in Operations Research and Management Science, Volume 8: Network Routing*, Elsevier, Amsterdam, 35-140.

Du Merle, O., D. Villeneuve, J. Desrosiers, P. Hansen. 1999. Stabilized column generation. *Discrete Mathematics* **194** 229-237.

Ehrgott, M., D. M. Ryan. 2002. Constructing robust crew schedules with bicriteria optimization. *Journal of Multicriteria Decision Analysis* **11** 139-150.

Ford, L. R., D. R. Fulkerson. 1958. A suggested computation for the maximal multicommodity network flows. *Management Science* **5** 97-101.

Gilmore, P. C., R. E. Gomory. 1961. A linear programming approach to the cutting stock problem. *Operations Research* **9** 849-859.

Girard A., Sansó, B. 1998. Multicommodity flow models, failure propagation, and reliable loss network design. *IEEE/ACM Transactions on Networking* **6** 82-93.

Holmberg, K., Yuan, D. 2003. A multicommodity network-flow problem with side constraints on paths solved by column generation. *INFORMS Journal on Computing* **15** 42-57.

ILOG 2002. ILOG CPLEX 8.0 Reference Manual.

Johnson, E. L. 1989. Modeling and strong linear programs for mixed integer programming. S. W. Wallace ed. *Algorithms and Model Formulations in Mathematical Programming*, Springer-Verlag, 1-43.

Kleywegt A. J., A. Shapiro, T. Homem-de-Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* **12** 479-502.

Laporte, G., F. V. Louveaux. 1993. The integer L-shaped method for stochastic integer programs with complete resource. *Operations Research Letters* **13** 133–142.

Laporte, G., F. V. Louveaux, L. Van Hamme. 1994. Exact solution to a location problem with stochastic demands. *Transportation Science* **28** 95–103.

Linderoth, J., S. J. Wright. 2002. Decomposition algorithms for stochastic programming on a computational grid. Optimization Technical Report 02-07, Computer Sciences Department, University of Wisconsin-Madison.

Lorena, L. A. N., E. L. F. Senne. 2004. A column generation approach to capacitated p-median problems. *Computers and Operations Research* **31** 863-876.

Lougee-Heimer, R. 2003. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development* **47** 57-66.

Louveaux F. V., D. Peeters. 1992. A dual-based procedure for a stochastic facility location. *Operations Research* **40** 564-573.

Lübbecke, M. E., J. Desrosiers. 2002. Selected topics in column generation. Les Cahiers de GERAD G-2002-64, Group for Research in Decision Analysis, Montreal, Canada. http://www.optimizationonline.org/DB_FILE/2002/12/580.pdf (accessed July 2004)

Lulli, G., S. Sen. 2004. A branch-and-price algorithm for multi-stage stochastic integer programming with application to stochastic batch-size problems. *Management Science*, to appear.

Mak, W.-K., D. P. Morton, R. K. Wood. 1999. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters* **24** 47-56.

Papagiannaki, K., S. Moon, C. Fraleigh, P. Thiran, C. Diot. 2003. Measurement and analysis of single-hop delay on an IP backbone network. *IEEE Journal on Selected Areas in Communications* **21** 908-921.

Ralphs, T. K., L. Ladanyi. 2001. *COIN/BCP User's Manual*. http://www-124.ibm.com/developerworks/opensource/coin/presentations/bcp-man.pdf

Ribeiro C. C., F. Soumis. 1994. A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research* **42** 41-52.

Ryan, D.M., B.A. Foster. 1981. An integer programming approach to scheduling. A.Wren, ed. *Computer Scheduling of Public Transport, Urban Passenger Vehicle and Crew Scheduling*. North Holland, Amsterdam. 269–280.

Savelsbergh, M. W. P. 1997. A branch-and-price algorithm for the generalized assignment problem. *Operations Research* **45** 831-841.

Sen, S., J. L. Higle. 2000. The C^3 theorem and a D^2 algorithm for large scale stochastic integer programming: Set convexification. *Stochastic Programming E-Print Series*. (http://dochost.rz.hu-berlin.de/speps).

Shiina T., J. R. Birge. 2004. Stochastic unit commitment problem. *International Transactions in Operational Research* **11** 19-32.

Singh, K., A. Philpott, R. K. Wood. 2004. Column generation for designing survivable electric power networks. Working paper, Dept. of Engineering Science, University of Auckland, Auckland, New Zealand.

Silva, E. F. 2004. Improving branch-and-price algorithms and applying them to stochastic programs. PhD Dissertation, Naval Postgraduate School, Monterey, CA, in preparation.

Spoerl, D., R. K. Wood. 2003. A stochastic generalized assignment problem. INFORMS Annual Meeting, Atlanta, GA, 19-22 Oct.

Teo, C-P., J. Shu. 2004. Warehouse-retailer network design problem. *Operations Research* **52** 396-408.

Vance P. H., C. Barnhart, E. L. Johnson, G. L. Nemhauser. 1997. Airline crew scheduling: A new formulation and decomposition algorithm. *Operations Research* **45** 188-200.

Walkup, D. W., R. J.-B. Wets. 1967. Stochastic programs with recourse. *SIAM Journal on Applied Mathematics* **15** 1299–1314.

Weingartner, H. M., D. N. Ness. 1967. Methods for the multidimensional 0/1 knapsack problem. *Operations Research* **15** 83-103.

Wets, R.J.-B. 1966. Programming under uncertainty: the complete problem. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **4** 316–339.

Wolsey, L. A. 1998. *Integer Programming*. John Wiley & Sons, New York.

Zhou, J., B. Liu. 2003. New stochastic models for capacitated location-allocation problem. *Computers and Industrial Engineering* **45** 111-125.